LABORATORIO 9

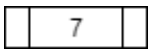
Apuntadores y Listas dinámicas - Básico

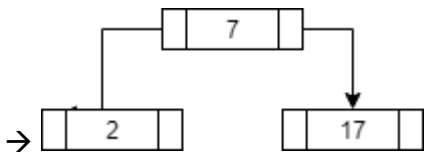
1. Implementa la función **esta** que, dado un valor entero N y una lista dinámica con valores que pueden no estar ordenados, diga si el entero está o no en la lista.
2. Implementa la función **posición** que, dado un valor entero X y una lista dinámica cuyos valores pueden no estar ordenados, diga en qué posición se encuentra X en la lista. Si dicho entero estuviera repetido, bastará con devolver la posición de la primera de sus apariciones (la primera posición es 1); y en caso de no encontrarse en el vector, se devolverá el valor *Integer'Last*.
3. Implementa la función **Insertar_delante** que, dada una lista dinámica y un entero, modifique dicha lista anteponiendo el entero a la lista que había. Por ejemplo, dados (1 2 3 4 5) y 0, debe devolver (0 1 2 3 4 5).
4. Implementa la función **Insertar_detrás** que, dada una lista dinámica y un entero, modifique dicha lista añadiendo al final de la lista el entero. Por ejemplo, dados (1 2 3 4 5) y 6, debe devolver (1 2 3 4 5 6).
5. Implementa la función **Insertar_elemento_en_pos** que, dada una lista dinámica, un entero, y una posición, modifique dicha lista incluyendo el entero en la lista que había. Se considera que el primer elemento es la posición 1. Todas las posiciones son válidas: si la posición de inserción es previa al primer elemento, se añade por delante de la lista; Si la posición es mayor que la posición del último se añade al final; y si es intermedia, se intercala para que el elemento añadido ocupe la posición indicada. Por ejemplo, añadir 0 en la posición -2500 de (1 2 3 4 5), debe devolver (0 1 2 3 4 5), porque -2500 es una posición anterior a la 1. Si la fuera añadir 1000 en la posición 1000 de (1 2 3 4 5) debería devolver (1 2 3 4 5 1000). Y si hubiera que añadir 25 en la posición 3 de (10 20 30 40 50), el resultado debería ser (10 20 25 30 40 50).
6. Implementa la función **Borrar** que, dada una lista y un entero N, modifica la lista eliminando de la lista la primera aparición del valor N. Si N no está en la lista, la lista se queda como está.
7. Implementa la función **Concatenar** que, dadas dos listas dinámicas L1 y L2, las modifica para construir una lista dinámica que tiene los nodos de L2 a continuación de los de L1. No se puede usar **new** en este ejercicio).
8. Implementa el procedimiento **Invertir** que, dada una lista dinámica de enteros, la modifique para que los enteros aparezcan en orden inverso. O sea, que si se le pasa la lista (1 2 3 4 5) al procedimiento, éste debe devolver (5 4 3 2 1).
9. Implementa el procedimiento **Simplificar** que, dada una lista dinámica de puntos, la modifique eliminando los puntos que estén a una distancia menor a 0.001 en ambas coordenadas. Los puntos que queden en la lista deben mantener el orden de la lista original y los puntos inicial y final siempre se habrán de mantener. Se penalizará el uso de listas auxiliares en la resolución del problema.

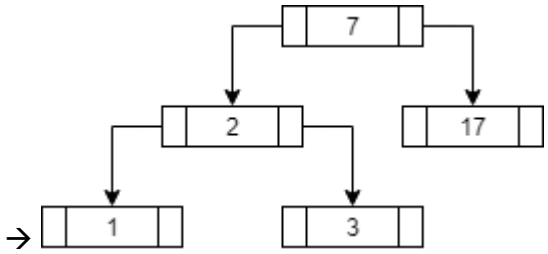
LABORATORIO 9

Apuntadores y Listas dinámicas – Avanzado

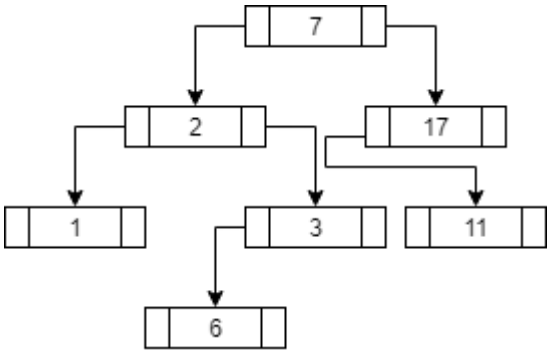
1. Implementa la función **Media** que, dada una lista de enteros L, devuelva la media aritmética de los enteros de la lista. Si la lista es vacía, se devolverá Float'Last.
2. Implementa el procedimiento **Máximo** que, dada una lista de enteros L, devuelva el máximo valor de los de esa lista y la posición en la que se encuentra en la lista (el primer elemento de la lista está en la posición 1). Si la lista es vacía se devolverá Integer'First y 0, respectivamente.
3. Implementa el procedimiento **Insertar** que, a partir de una lista dinámica de enteros ordenada ascendentemente y un número natural N, lo inserte en la posición que le corresponde (o sea, manteniendo el orden de los elementos de la lista).
4. Implementa la función **Clonar_y_Concatenar** que, dadas dos listas dinámicas L1 y L2, cree una nueva lista L, diferente de las anteriores que tenga una copia de todos los elementos de L1 y, a continuación, una copia de todos los elementos de L2.
5. Implementa el procedimiento **Borrar** que, dada una lista dinámica y un entero N, modifique la lista eliminando de la lista todas las apariciones del entero N en la lista.
6. Implementa el procedimiento **Intersección** que, dadas dos listas dinámicas L1 y L2, ordenadas ascendentemente, reorganice los elementos en otras dos listas, comunes y no_comunes, también ordenadas ascendentemente, de manera que la primera contenga los elementos que están en ambas listas (dos veces) y la segunda los que solo están en una de ellas. L1 y L2 después de la llamada al procedimiento quedan deshechas. En la implementación de este subprograma, no puede haber instrucciones new.
7. Implementa la función **Son_iguales** que dadas dos listas dinámicas de enteros, ordenadas o no, indique si ambas listas contienen el mismo número de elementos y en el mismo orden.
8. Implementa el procedimiento **Simplificar** que, a partir de una lista de puntos no vacía L, la modifique para que solo tenga un cuarto de los puntos, respetando uno sí y tres no del original. Los puntos que se queden deben mantener el orden de la lista original y los puntos inicial y final siempre se habrán de mantener. También se devolverá el número de puntos resultantes en la lista resultado. (Es el mismo ejercicio que el laboratorio 7 avanzado, pero con listas dinámicas).
9. Implementa el procedimiento **Crear_arbol_binario** que, dado un vector de enteros, cree un árbol binario de la siguiente manera. Se mira el valor del entero de la raíz. Si el valor a insertar es menor que él, se introduce en el subárbol izquierdo y si no en el derecho (si es igual, se ignora). Si no hay subárbol izquierdo o derecho, se crea un nodo con el elemento a insertar y si no, se decide en qué rama hay que introducir el nuevo elemento de la misma manera que antes (menores a la izquierda y mayores a la derecha). Por ejemplo, si el vector contiene los valores (7 2 17 1 3 11 6 16 4 5 18 8). Del árbol vacío se crearía uno con solo el 7. Luego, el 17 se pondría colgando de la rama derecha (porque 17>7) y el 2 colgando de la rama izquierda (porque 2<7). Con los siguientes números se haría parecido: el 1 va por la rama izquierda del 7 y la izquierda del 2. El 3 seguiría la rama izquierda del 7 y la derecha del 2. Y así sucesivamente.



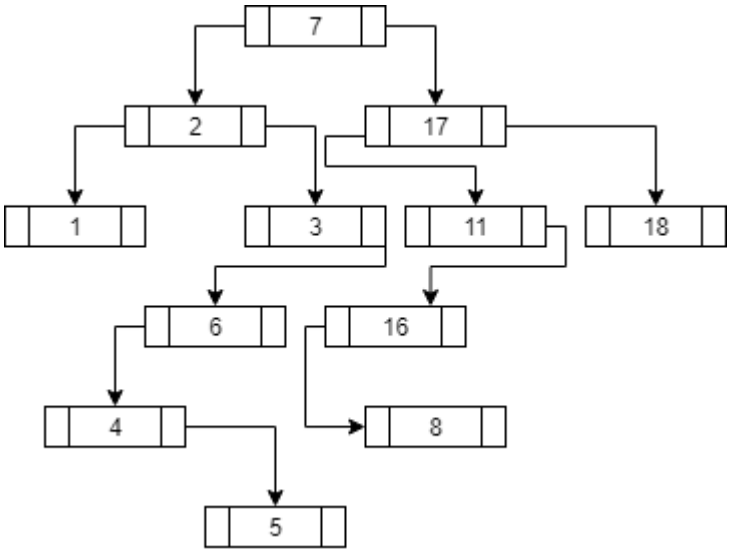



7
2
17
1
3

→


7
2
17
1
3
11
6

→


7
2
17
1
3
11
18
6
16
4
8
5

→